ENQPY™

THE STAKES STATEMENT

The master public-facing messaging document

FINAL · PUBLIC MESSAGING ANCHOR

Anchor for site, video, press, outreach · NQP LLC · Paul McGough · Rev 5.0 (Canonical Release) · June 2026

Encrypted today. Unreadable forever.

Capture the ciphertext. Store it for a century. Bring any computer ever built — quantum included. It stays unreadable. The information needed to read it isn't in the ciphertext. It was never there to find.

This is a formally proved mathematical claim, with the proof and the code public for anyone to check.

For seventy-five years, secrecy has meant "hard to break."

Hard is not impossible. Anything encrypted today can be copied, stored, and attacked tomorrow — message by message, record by record, file by file. The world's captured ciphertext is sitting on ice, waiting for the machines that can read it. That is the threat the whole industry now calls harvest-now, decrypt-later: store it today, break it later.

Enqpy™ ends "decrypt later."

Enqpy™ is a new encryption cipher with a formal proof: captured ciphertext, on its own, never becomes readable. Not with more time. Not with more captured ciphertext. Not with quantum computers.

The reason is structural. The information needed to recover the message is not in the ciphertext — not hidden, not scrambled, not waiting to be found. Not there. The proof establishes this as a property of the math itself.

We call it **harvest-proof confidentiality**: captured ciphertext stays unreadable because there is nothing in it to wait for.

The One-Time Pad reached this kind of secrecy in 1949 — and made it useless, by demanding a key as long as every message. Enqpy™ reaches non-vanishing ciphertext-only uncertainty with a finite, reusable key. That is the result cryptography has wanted for seventy-five years: ciphertext-only secrecy that leans on nothing being hard to compute, made practical.

The old tradeoff is gone.

Provable security used to mean impractical security — massive keys, slow systems, narrow use. “Pick two of three: secure, fast, flexible.” Enqpy™ broke the rule. The same algorithm that carries the proof is a small, fast, portable stream cipher with zero ciphertext expansion, scalable into hardware, deployable from 8-bit microcontrollers to 64-bit servers. Under the published benchmark conditions, the reference runs at about 900 MB/s in portable C and about 6,400 MB/s on FPGA.

The proof wasn’t the constraint. The proof was the prize.

The proof is public. The code is public.

The formal proof is published — at enqpy.com, in the GitHub repository, with permanent archival deposits (Zenodo, Software Heritage, Internet Archive). The C reference implementation is open source under Apache 2.0. The test vectors reproduce. The 84-assertion self-test verifies any conforming build against the published vectors. Clone it, run it, compare the outputs, read the proof. Nothing is hidden.

And one more thing you can check: we own the patents, and we pledged — irrevocably, dated, successor-binding — never to assert them against the open world. Read the pledge, date it, confirm it binds whoever comes next. We proved it; you can check the proof. We pledged it; you can check the pledge.

How you run it.

Run Enqpy™ the way it ships: each record under its own fresh key, authenticated — the same per-record hygiene you already apply with AES-GCM’s fresh nonce. Under that discipline, captured ciphertext does not expire. The harvest-now-decrypt-later threat to your users is closed by the proof, and the post-quantum migration treadmill ends: you are no longer replacing one hardness assumption with the next.

That is the difference.

Say it correctly.

For journalists, partners, and anyone explaining Enqpy™.

One sentence. Enqpy™ is a practical finite-key encryption cipher whose published proof establishes that captured ciphertext, on its own, stays permanently unreadable — no amount of time, captured data, computation, or quantum hardware reaches the plaintext.

Plain English. Today’s encryption makes secrets hard to recover. Enqpy™ makes the captured ciphertext, by itself, not contain enough to recover them at all — no matter how long an attacker waits or how much compute they bring.

Practical reality. Enqpy™ isn't slower or heavier than what it replaces: about 900 MB/s in portable C, about 6,400 MB/s on FPGA, zero ciphertext expansion, from 8-bit microcontrollers to 64-bit servers. The proved cipher is the practical one.

For cryptographers. The proof is a non-vanishing ciphertext-only equivocation, motivated by Shannon's Ideal-System target: from captured ciphertext the message stays uncertain to at least 2^{128} equally consistent plaintexts (uniform posterior at HIGH) and the key to a 2-bit floor, with no computational-hardness assumption — so no amount of compute, quantum included, touches it. The proof also tells you exactly where the guarantee ends: reuse one key across known plaintext and exposure transfers at about two fully-known windows. That is why each record gets its own fresh key, authenticated (encrypt-then-MAC). Run the code. Read the proof. Find the line yourself.

How the rights work. Enqpy™ is patent-pledged for open public use and licensed at the edge. NQP LLC owns the patents and has published a dated, irrevocable, successor-binding covenant not to assert them against the cipher as defined in the covenant — study, research, evaluation, open implementation, standards-track adoption, and commercial deployment at any scale, with no permission and no fee. What NQP licenses is the edge: certification and the Enqpy™ mark, the high-performance implementation, support, integration, and indemnity.

What to do.

Read the Plain-Language Proof Guide — no equations required. Clone the GitHub repository. Run the self-tests. Read the proof if you want the formal argument.

Run the code. Check the benchmarks. Inspect the proof. Find out.

APPENDIX A

HOMEPAGE HERO

Above-the-fold copy, derived from the Stakes Statement

Encrypted today. Unreadable forever.

Capture the ciphertext. Store it for a century. Bring any computer ever built — quantum included. It stays unreadable. The information needed to read it isn't in the ciphertext.

Proof public. Code public. Test vectors public.

[Start Here] · [See the Code] · [Read the Proof]

— End of Enqpy™ Stakes Statement (Final) —